Pradham Mummaleti

SE20UARI116

CSM Assignment

Pre-trained Model:

Vision Encoder Model: timesformer-base-finetuned-k600

Text Decoder Model: GPT - 2

Encoder is based on novel architecture called TimeSformer.

- Uses self – attention mechanisms to directly learn spatiotemporal features from patches from video frames.
- Challenges traditional 3D CNNs by:
    - Being significantly faster to train and requires less data
    - Can be applied to longer videos than traditional CNNs due to its lower computational cost.

Limitations of TimeSformer:

- Works well for larger datasets, but more investigation is required for smaller datasets.
- May struggle for videos containing intricate details
- CNNs offer more interpretability compared TimeSformer.
- Scalability for longer videos such as moives remains a question.

Limitation of GPT-2:

- While generating longer than a couple paragraphs, there's an occasional non-sequiter, and longer samples of GPT – 2 tend to stray off topic.

Test:-

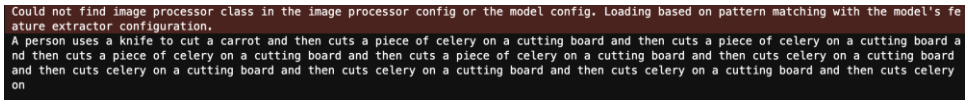
Test Data Clip: Person using a knife to cut celery and carrots in different styles (6 min long video)

Output:

```
Could not find image processor class in the image processor config or the model config. Loading based on pattern matching with the model's fe
ature extractor configuration.
A person is demonstrating how to cut celery in half and how to cut it in half with a knife on a cutting board and then cut it in half with a
knife on a cutting board and then cut it in half with a knife on a cutting board and then cut it in half with a knife on a cutting board and
then cut it in half with a knife on a cutting board and then cut it in half with a knife on a cutting board and then cut it in half with a kn
ife
```

This output comes from extending the minimum number of words (to 80 words) in the caption. The model struggles capturing the main jist of the video. This with the beam number = 8.

Now let's try number of beams = 15.



More context is given but it does not consider the several types of cutting techniques demonstrated in the video.

One main drawback is that it does not capture audio information from the video as well. For this we would want to look at a model such as VALOR.

VALOR: Vision-Audio-language Omni-Perceptioin Pretraining

Architecture:

Encoders:

- Vision Encoder
- Audio Encoder
- Lanaguage Encoder

Decoder:

- Conditional Text Generation: This component generates the caption with the learned multimodal representation of the input. Can be trained on different objectives, such as maximising the likelihood of the correct caption or minimizing reconstruction errors.

Strengths:

- Combines visual, auditory and textual cues to enhance video understanding.
- Increased captioning accuracy
- Avoids contradictions or inconsistencies between what is seen, heard, or described.
- Can be fine-tuned for specific tasks such as video question answering or multimodal summarization.

I have incorporated a model in the code, which can work along with the video summarizer to transcribe the model.

The transcipt can then be put into a text summarizer and from that we get a captioned audio along with a captioned video.

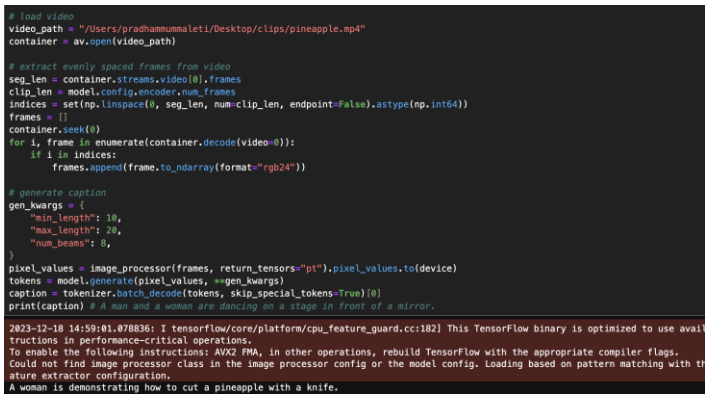
Model for text summarization: Fine-Tuned T5 Small for Text Summarization

The **Fine-Tuned T5 Small** is a variant of the T5 transformer model, designed for the task of text summarization. It is adapted and fine-tuned to generate concise and coherent summaries of input text.

Limitation: Great for Text Summarization, mediocre in the other language processing tasks

Example: Pineapple cutting video

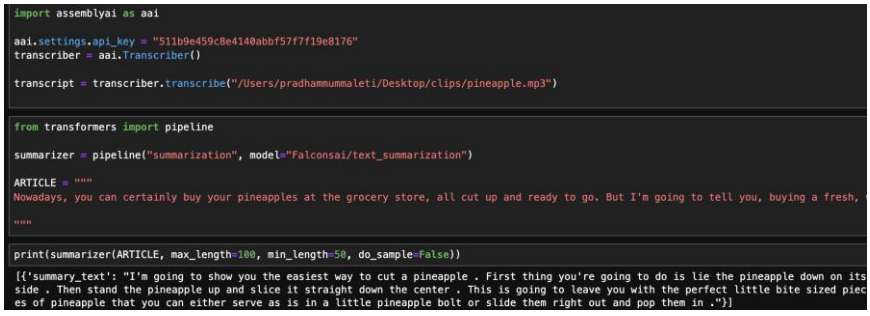
Here is the video captioning from the pretrained model.



```
# load video
video_path = "/Users/pradhammummaleti/Desktop/clips/pineapple.mp4"
container = av.open(video_path)

# extract evenly spaced frames from video
seg_len = container.streams.video[0].frames
clip_len = model.config.encoder.num_frames
indices = set(np.linspace(0, seg_len, num=clip_len, endpoint=False).astype(np.int64))
frames = []
container.seek(0)
for i, frame in enumerate(container.decode(video=0)):
    if i in indices:
        frames.append(frame.to_ndarray(format="rgb24"))

# generate caption
gen_kwargs = {
    "min_length": 10,
    "max_length": 20,
    "num_beams": 8,
}
pixel_values = image_processor(frames, return_tensors="pt").pixel_values.to(device)
tokens = model.generate(pixel_values, **gen_kwargs)
caption = tokenizer.batch_decode(tokens, skip_special_tokens=True)[0]
print(caption) # A man and a woman are dancing on a stage in front of a mirror.
```

```
2023-12-18 14:59:01.078836: I tensorflow/core/platform/cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use avail
tructions in performance-critical operations.
To enable the following instructions: AVX2 FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Could not find image processor class in the image processor config or the model config. Loading based on pattern matching with th
ature extractor configuration.
A woman is demonstrating how to cut a pineapple with a knife.
```

Taking the audio file from the mp4 video, we can use a pretrained model to extract the transcript from the audio file. Then, we use the text summarizer.

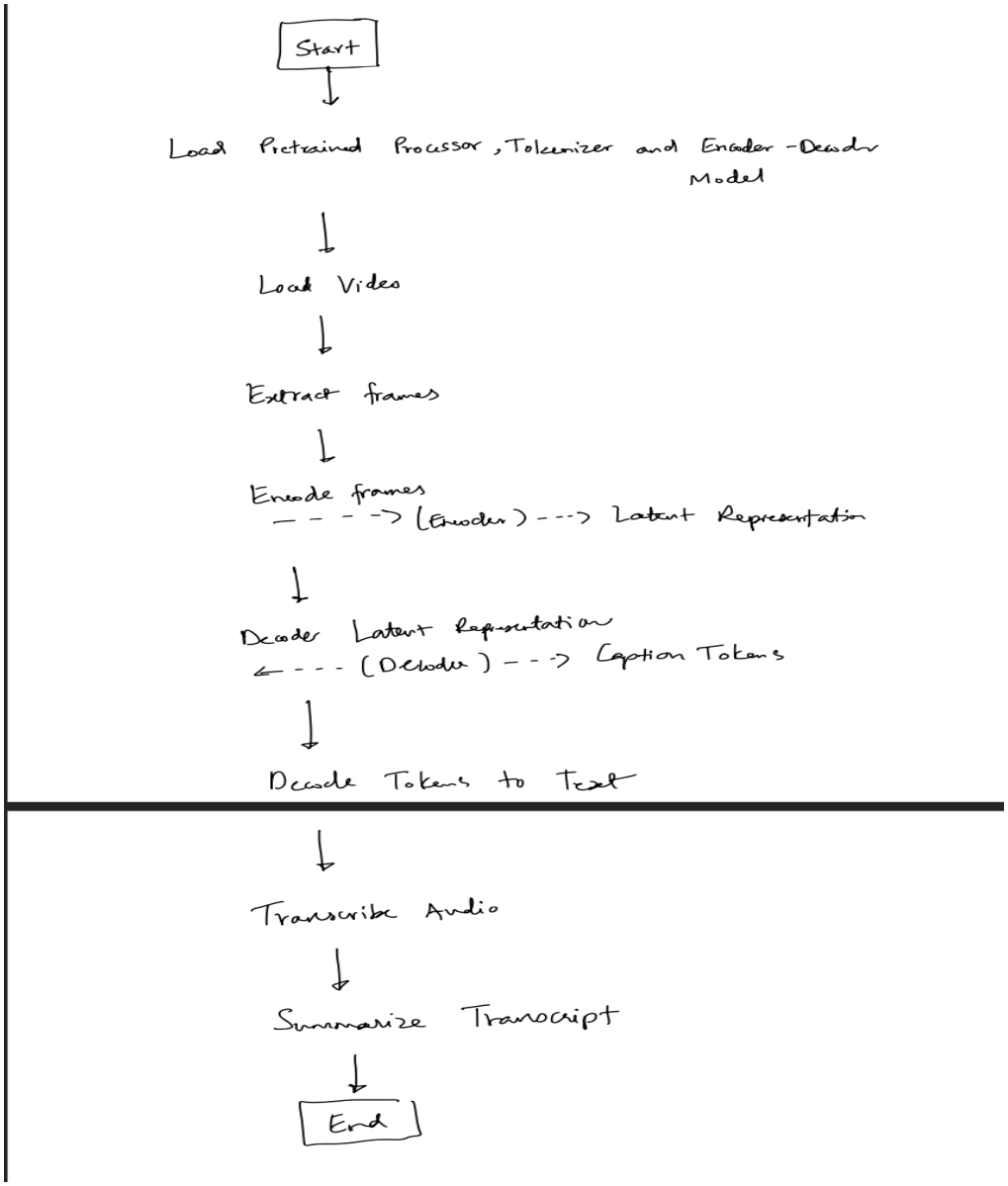


```
import assemblyai as aai

aai.settings.api_key = "511b9e459c8e4140abbf57f7f19e8176"
transcriber = aai.Transcriber()

transcript = transcriber.transcribe("/Users/pradhammummaleti/Desktop/clips/pineapple.mp3")
```

```
from transformers import pipeline

summarizer = pipeline("summarization", model="Falconsai/text_summarization")

ARTICLE = """
Nowadays, you can certainly buy your pineapples at the grocery store, all cut up and ready to go. But I'm going to tell you, buying a fresh,

"""

print(summarizer(ARTICLE, max_length=100, min_length=50, do_sample=False))
```

```
[{'summary_text': "I'm going to show you the easiest way to cut a pineapple . First thing you're going to do is lie the pineapple down on its
side . Then stand the pineapple up and slice it straight down the center . This is going to leave you with the perfect little bite sized piec
es of pineapple that you can either serve as is in a little pineapple bolt or slide them right out and pop them in ."}]
```

Flowchart:


Start
Load Pretrained Processor, Tokenizer and Encoder-Decoder Model
Load Video
Extract frames
Encode frames
---> (Encoder) ---> Latent Representation
Decoder Latent Representation
<--- (Decoder) ---> Caption Tokens
Decode Tokens to Text
Transcribe Audio
Summarize Transcript
End